Mutex locked twice

Name: Mutex locked twice

Description: Calling the lock method of a mutex twice in succession might cause a deadlock.

ID: cpp/twice-locked

Kind: problem

Severity: error

Precision: low

Mutexes come in two flavors: recursive and non-recursive. For example, the C++ mutex library provides both `std::mutex` and `std::recursive_mutex`. A non-recursive mutex cannot be locked until it has been unlocked by its previous owner, even if it is already owned by the current thread. Deadlock is often caused by a thread attempting to lock the same mutex twice, usually in a recursive algorithm.

Recommendation

If a recursive method needs to acquire a lock, then split it into two methods. The first method is public and is responsible for locking and unlocking the mutex. It delegates the rest of the work to a second private method. The second method does not need to lock or unlock the mutex because that is done by the first method.

Example

In this example, the method `f` is recursive, so it causes a deadlock by attempting to lock the mutex twice.

```
1  class C {
2    std::mutex mutex;
3  public:
4    // BAD: recursion causes deadlock.
5    int f(int n) {
6      mutex.lock();
7      int result = (n == 0) ? 1 : n*f(n-1);
8      mutex.unlock();
9      return result;
10   }
11 };
```

In this second example, the recursion is delegated to an internal method so the mutex is only locked once.

```
1  class C {
2    std::mutex mutex;
3    int f_impl(int n) {
4      return (n == 0) ? 1 : n*f_impl(n-1);
5    }
6  public:
7    // GOOD: recursion is delegated to f_impl.
8    int f(int n) {
9      mutex.lock();
10     int result = f_impl(n);
11     mutex.unlock();
12     return result;
13   }
14 };
```

References

- Common Weakness Enumeration: CWE-764.

- Common Weakness Enumeration: CWE-833.